Running User-provided Virtual Machines in Batch-oriented Computing Clusters

Vítor Oliveira, António Pina, André Rocha
Departamento de Informática, Universidade do Minho
Campus de Gualtar, Braga, Portugal
{vspo,amp,arocha}@di.uminho.pt

Abstract

The use of virtualization in HPC clusters can provide rich software environments, application isolation and efficient workload management mechanisms, but system-level virtualization introduces a software layer on the computing nodes that reduces performance and inhibits the direct use of hardware devices. We present an unobtrusive user-level platform to execute virtual machines inside batch jobs that does not handicap the computing cluster's ability to execute the most demanding applications. A per-user platform uses a static mode in which the VMs run entirely within the resources of a single batch job and a dynamic mode in which the VMs navigate at runtime between the continuously allocated jobs node time-slots. In the dynamic mode fault-tolerant system agents are integrated using group communication to control the system, to execute user commands and to implement user-defined scheduling policies. In our tests compute intensive applications suffered negligible performance overhead compared to the native configuration, but the user-mode network overlay introduced a significant penalty on the more taxing networked applications.

1. Introduction

Virtualization has become pervasive in many areas of computing, but it is still avoided in HPC systems in order to achieve maximum performance and to adapt to complex environments that use diverse computing and networking elements [1,2].

Several techniques have been developed to minimize the overhead of virtualization in HPC, which is also offset by the effective use of the resources enabled by virtualization [1,3]. In the computing nodes, in particular, it can be used to construct i) rich, purpose adapted software configurations, ii) constant application environments not dependent on the underlying hardware and iii) flexible and efficient workload management systems.

The User Domains (UD) platform we introduce executes user provided virtual machines (VMs) in batch-oriented cluster computing nodes allocated for typical user jobs. It offers an unobtrusive execution environment that is only deployed on the nodes when required to avoid permanent software that might limit the nodes' ability to execute typical cluster applications at peak performance. The system can function in two distinct execution modes. In the static mode the job collects several time-slots in which VMs are immediately ran; the setup is fixed at start-up and VMs must execute within the time slots allocated for the job. In the dynamic mode more flexible configurations are possible with variable time-slot allocation; the jobs starts a proxy to represent the user platform in the computing nodes. The proxy agents cooperate to build an overlay infrastructure that is foundational to execute, suspend and move VMs on user request. It allows migration to be explored for long running VMs, as they can be transported from node to node according to the availability of time slots in the batch system. A user provided management function can trigger VM transfer between time slots to provide a quasi-continuous mode for service-oriented workloads.

The platform is part of a broader project to deploy replicated VMs on a Grid, so it was largely influenced by the need to easily exploit several clusters in distinct administrative authorities. To this end only user mode execution is used and network overlays control network access and maintain safe and integrated user environments. The focus was on the ability of running coherently a few VMs in existing batch clusters. The design does not scale well to a large number of machines per user, limited mainly by the network overlays. But larger deployments are already served by a number of specialized Cloud infrastructures.

The main issues addressed to run VMs in cluster batch jobs are presented in section II, the support for dynamic VM configurations in section III and the system performance in section IV, followed by the related work and the conclusions sections.

2. Executing VMs in cluster batch jobs

The execution of VMs is more demanding than that of typical HPC applications because the system must not only execute the intended application but also the environment that contains it [7]. In a conventional batch-oriented computing cluster this is further complicated by our need to run in user mode and to support vary dissimilar environments in the VMs, which meant that no special OS or device drivers should be mandatory in the host or in the VMs.

A non-invasive virtualization layer becomes fundamental to ensure that: i) the cluster nodes can simultaneously execute typical cluster applications and VMs, ii) no significant changes to the software base are required, iii) there is no interference in the host system when virtualization is unused and iv) no administrative privileges are required. The main issues were: i) how to run the VMs in the cluster computing nodes and ii) how to network the VMs. Presently the cluster file system stores the VMs, but research on a fast and compact user-mode storage system is ongoing.

Fig. 1 presents an overview of the process of executing VMs in our system, including the dynamic mode presented in the next section.

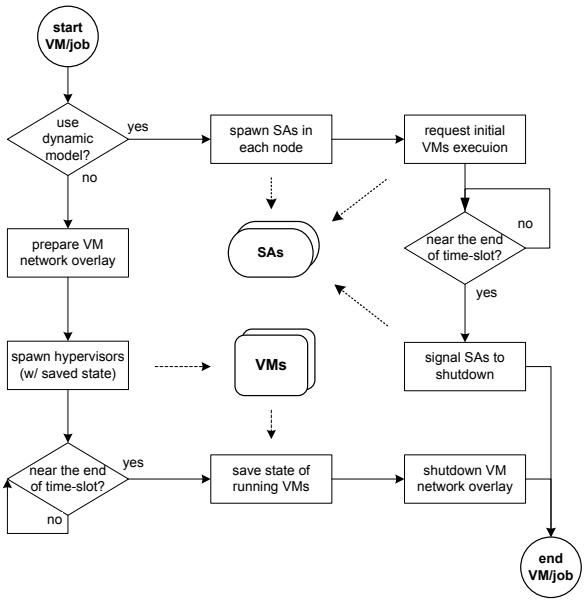

Fig. 1 Flow of VM execution

2.1. User-mode VM Execution

The key component to run VMs is the virtualization hypervisor, which provides an artificial environment that is multiplexed in the real hardware [5]. While system level hypervisors take full control of the host system and have a profound impact on the environment even when no VMs running, the QEMU [6] process level hypervisor was selected to encapsulate VMs in normal processes and let the host assume scheduling and resource allocation for VMs. QEMU is a full featured hypervisor/emulator that addresses the fundamental live migration of VMs and the support for hardware virtualization (with KVM) to offer a competitive level of performance. It also provides mechanisms to plug user-level network and disk interfaces to the VM safely and on demand.

The system starts the hypervisor, or an agent, when the time slots are requested. The VMs then starts up and the user can execute the applications and then shuts down the VMs before the job time-slot expires. If necessary, the state can persist between job executions to allow the execution of applications longer then the available time slot by snapshotting the VM.

2.2. User-mode Ethernet network overlay

A private Ethernet VM network overlay based on VDE [7] is deployed per user. The interface between this private network of VMs and the exterior is the responsibility of the Gateway component (see fig. 2) which handles: i) routing to and from the network, ii) masquerading of the internal addresses, iii) server and port publishing to allow inbound connections and iv) DHCP and DNS services for the VMs. In order to manage the network addressing, the internal DNS and DHCP services were adapted to support the dynamic inclusion of VMs using a static implementation of an addressing policy that rebuilds the missing data in case the services fail and are restarted in a new location.

While the use of NAT to masquerade out-bound VM network traffic effectively provides a layer of protection that isolates the internal nodes, it also limits the users ability to connect to the VM not only from nodes external to the cluster (outside the cluster gateway), but also from cluster nodes that have no direct access to the users network such as the login nodes from which the users connect. In order to support external connections a publishing mechanism was devised to map internal servers and ports on the network overlay gateway to which external processes connect. The port redirection allows reaching the IP ports of the VMs using a tunnel through a common ssh connection of a logged-in user session. The publishing of the server/port pairs required by each user VM is performed by the Gateway, which then tunnels the connections to the correct servers/ports. A Networker application can be executed on the nodes that require access to the VMs, such as the login nodes, to locate the Gateway and establish the connections. Unlike the internal Ethernet traffic, the migration of VMs is not transparent to the published connections.

3. Support for dynamic configurations

The dynamic execution mode provides a greater degree of control over the execution of the VMs, as users take advantage of resources dynamically acquired, instead of being limited to the resources of a single batch job. As each time slots ends, the user can submit additional jobs that will spawn the required user agents into the recently allocated computer resources, which will then join the run-time to continue the execution of the VMs. A monitoring function can act as a second level scheduler and resource manager for user specific requirements, acting when the user is disconnected from the system and is unable to manage the assigned time slots and grow or shrink the resource pool as desired. If the user does not possess enough resources in the system before the time-slot expires, the VMs will be suspended until the corresponding resources are available. With the ability to migrate VMs between the time-slots this allows the system to execute VMs for long periods and to adjust to changing conditions or respond to resource scarcity.

The run-time is composed of one or more agents, one per node, each capable of instantiating the services needed to execute user-provided VMs according to the user requests. This execution environment is unique for each user and includes a private reliable control network. It isolates the users VMs and presents an interface system to mediate the connection with the other components of the system.

3.1. An autonomous System Agent

The Systems Agents (SA) are proxies of the system in nodes assigned to the user, deployed by the batch system when a job starts and that execute until the end of the request time slot. These multi-threaded processes interface with each other over a dedicated overlay and connect to i) the local hypervisors to send requests to perform VM related operations, ii) the VM overlay network daemons and iii) the user interface to handle commands and status information.

The SAs orchestrate the tasks related to the management of the environment and to the user VMs, including to: i) set up the supporting software in the nodes, ii) link to the storage system, iii) spawn and monitor the user-mode hypervisor, iv) accept action requests for migrating the VMs between nodes and suspending a VM to disk, and v) monitor the behavior of the system. One SA (mc) is elected to centralize some support functions, such as to initiate, terminate and repair the VM network overlay, and to execute the user-defined function to monitor the system.

3.2. A reliable SA network overlay

Since the components required to support VM/jobs run entirely within time-slots allocated in the nodes, as each time-slot expires the associated components will be terminated and need to be restart elsewhere to keep the associated services available. In order to make coherent decentralized decisions it is essential to use an efficient and resilient method to detect failures of the various SAs in the system.

We chose to use a reliable group communication library (GCL) for inter agent communication and to manage agent membership [8]. The Virtual Synchrony approach greatly simplifies reaching agreements between the SAs and the management of dynamic SA configurations. Failures are consistently converted to group membership changes seen by all working nodes and global decisions can be made using local messages that circulate over the secure overlay. A consistent view of the system also eases the replication of VMs in less contained execution environments.

Even though a GCL has limited performance and scalability, the traffic between SAs is not exceedingly sensitive to latency and the number of nodes per user is expected to be well within typical GCL installations.

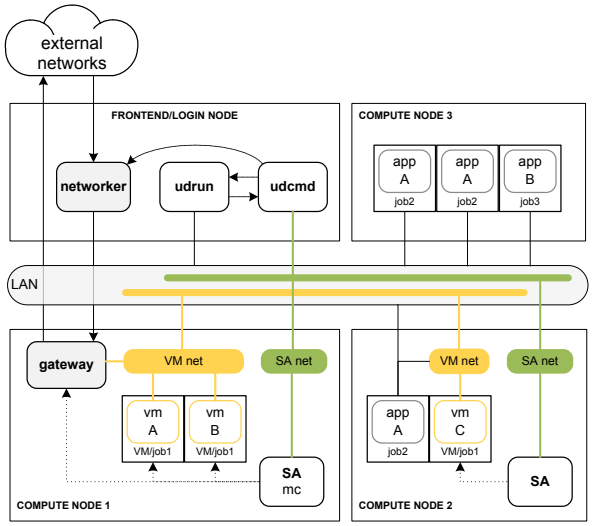

Fig. 2 System architecture

In fig. 2 the architecture of UD is presented with an example with three nodes running three jobs occupying 7 processing slots, with one job running 3 VMs. It includes the interface application **udrun** – to launch the system from the resource manager – and console **udcmd** – used to send commands to a running system.

4. Performance evaluation

The performance of the system was evaluated by benchmarks running on native hardware and on VMs located on the same host and on different hosts. The NAS Parallel Benchmarks (NPB) Multi-Zone edition [9] was used to simulate the calculations and data movement of computational fluid dynamics applications typical of HPC systems. The network bandwidth and latency were measured to evaluate the performance of the Ethernet network overlay.

The tests ran in two compute nodes with two hexa-core Intel Xeon X5650 processors per node, executing Linux OS kernel 2.6.18-194 and the QEMU/KVM 0.14.0 hypervisor. The node interconnect was Gigabit Ethernet and for VMs we tested both the user-mode VDE2-2.3.1 network overlay used in UD and the higher performance system mode TAP overlay. The 16 VMs were distributed evenly by socket and pinned to the assigned core, reserving the remaining cores for communication tasks. The median of 8 runs was used.

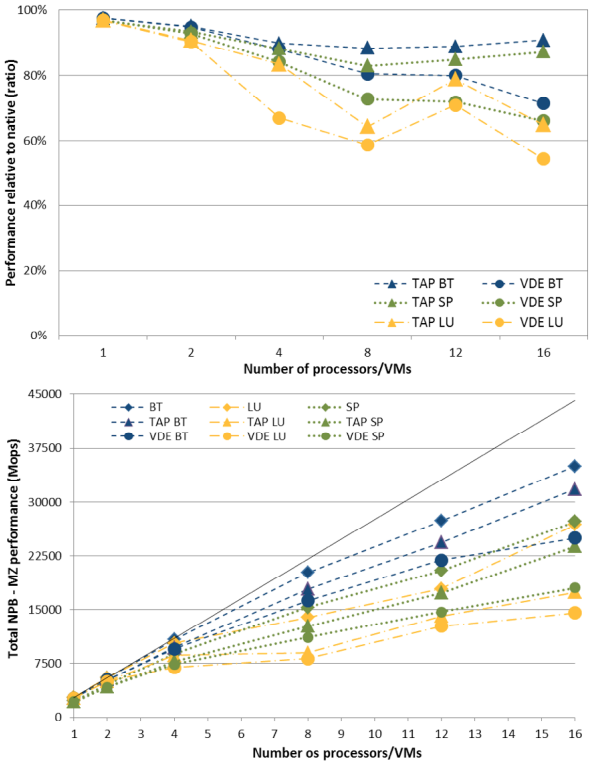

Fig. 3 Efficiency (top) and throughput (bottom) of NPB-MZ

4.1. NAS Parallel Benchmarks Multi-Zone

Figure 3 presents the NPB-MZ test results from 1 to 16 processors using two different views, representing the percentage of native performance VMs achieved (efficiency) and the total throughput. VMs achieved above 97% of the native hardware performance in all serial tests, close to the results in [4,5] for system level virtualization. It also shows a significant penalty for using the user-mode VDE Ethernet network instead of system-mode TAP. For the BT and SP tests that penalty raises steadily from 2% and 4% with 4 processors, to around 20% for 16 processors. As the processors increases the TAP configuration BT loses 12%, while LU loses up to 26%. The throughput always increased with the number of processors used.

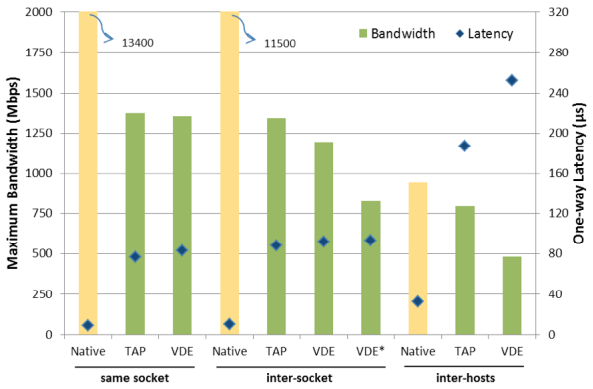

Fig. 4 Native mode, TAP and VDE network bandwidth and latency

4.2. Network bandwidth and latency

NetPIPE measured the network bandwidth and latency between two processes or VMs in the different locations. The inter-socket VDE test is split on whether the VDE switch is bound to the destination VM socket (VDE) or to the source VM processor socket (VDE*).

The same-socket inter VM bandwidth is almost 1.4Gbps, but it is nevertheless one order of magnitude lower than the bandwidth between two processes. The inter-socket bandwidth for VDE degrades to 825Mbps when the switch is not on the socket of the destination VM. Between nodes using GbE the TAP VMs reached almost 800Mbps, 15% away from the native bandwidth of 944Mbps, while only 50% of the native bandwidth is attained by VDE. VDE and TAP latencies between VMs in the same node are around 90us, slightly lower when VMs are on the same socket (around 80us), but the penalty increases between different nodes, with TAP doubling and VDE almost tripling native latency.

As expected, between VMs the network latency was higher and the bandwidth was lower than between processes. The network card virtualization makes the difference to native unsurprising, but we believe that two limitations in VDE contributed particularly to its increased overhead: i) the lack of parallelism in the VDE switch that hinders the performance when nodes communicate simultaneously and ii) the delay induced by inefficient VDE plugs and switch interconnections.

5. Related work

Xen, KVM and QEMU are used in [10] to deploy VMs in a cluster environment benefiting from live migration technology and load balancing with two-different scheduling levels: between VMs in the same physical node and between the physical nodes in the cluster. In [11] VMs are used as a resource provisioning mechanism for infrastructure-as-a-service (IaaS) clouds. In [12] PBS is used to launch on demand pre-defined VMs images in which user jobs can run. VMs also lead to the extension of scheduling [13] to include the automatic deployment of VMs before the application launching time and the decomposition of job submission in two phases: i) the deployment of the VMs and ii) the effective application execution using the standard MOAB capabilities. To provide support for specific user workloads in many virtualized environments, in [14] virtualization is used for transparent, on demand launching of VMs for running new jobs to form a MOSIX multi-cluster environment. In [15] a distributed peer-to-peer self-organizing and resilient environment is built by software running inside a VM, which can run in several hypervisors.

6. Conclusions

In this paper we presented a platform to provision, manage and execute VMs entirely in user mode on batch oriented clusters using: i) a static mode in which the VMs run using the resources of a single batch job and ii) a dynamic mode in which the VMs can navigate at runtime between nodes and time-slots allocated to several jobs. Without administrator intervention it explores features of virtualization, including OS customization, performance isolation, check pointing and migration and enables typical cluster jobs to run without incurring any performance degradation in the same nodes. The dynamic mode uses reliable group communication to integrate the fault-tolerant system agents, to allow more complex scenarios to be built using either direct user commands or automatic scheduling decisions made by a user functions. The dynamic mode is the foundation for advanced scheduling policies that support continuous service oriented workloads and that improve system-wide cluster resource utilization.

The experiments showed that serial applications run in VMs close to the native performance. The overhead in parallel applications is dependent on their communication pattern, but notwithstanding the verified network bandwidth and latency degradation the NPB-MZ benchmarks achieved 70% to 80% of the native speed up to 12 processors.

Finally, the platform presented embraces the challenges for an HPC virtualization solution [1]: it deploys the hypervisor as needed to reduce its footprint; it runs user-supplied VMs to support many virtual systems environments; a GCL makes SAs resilient and highly available; and a user-defined scheduler enables advanced management solutions.

7. References

[1] Vallee, G. et. al., "System-Level Virtualization for High Performance Computing", Proc. 16th Euromicro Conf. on Parallel, Distributed and Network-Based Processing, 2008.

[2] R. Iyer, "DatacenteronChip Architectures Terascale Opportunities and Challenges," Intel Technology Journal, vol. 11, Aug. 2007.

[3] W. Huang, J. Liu, B. Abali, and D.K. Panda, "A case for high performance computing with virtual machines," Proc. of the 20th annual Int. Conf. on Supercomputing, 2006.

[4] M. A. Murphy and S. Goasguen. "Virtual Organization Clusters: Self-Provisioned Clouds on the Grid," submitted to Elsevier Journal of Future Generation Computer Systems special issue on Cloud Computing, 2010.

[5] K. Adams and O. Agesen, "A comparison of software and hardware techniques for x86 virtualization," ACM SIGARCH Computer Architecture News, vol. 34, Oct. 2006.

[6] Fabrice Bellard. "QEMU, a Fast and Portable Dynamic Translator", In Proceedings of the 2005 USENIX Annual Technical Conference, April 2005.

[7] M. Goldweber and R. Davoli., "VDE: an emulation environment for supporting computer networking courses". In Proc. of the 13th annual Conf. on Innovation and Technology in Computer Science Education, 2008.

[8] Amir, Y. Stanton, J. "The spread wide area group communication system". TR CNDS-98-4, The Johns Hopkins University, 1998.

[9] Rob F. Van der Wijngaart and Haoqiang Jin. "Nas parallel benchmarks, multi-zone versions", Technical Report NAS-03-010, July 2003.

[10] Z. Wang, C. Weng, "A resource management mechanism and its impl. for virtual machines", Proc. of 2nd Int. Workshop on Systems and Virtualization Management: Standards and New Technologies, Germany, 2008.

[11] B. Sotomayor, R. S. Montero, I. M. Llorente, I. Foster, "Resource Leasing and the Art of Suspending Virtual Machines", High Performance Computing and Communications - HPCC, pp.59-68, 2009.

[12] R Espadamala, M Rodriguez, C Neissner, "Virtual machines over PBS", HEPiX Workshop, Spring 2010.

[13] W. Emeneker, D. Jackson, J. Butikofer, and D. Stanzione, "Dynamic Virtual Clustering with Xen and Moab", in Proc. ISPA Workshops, 2006, pp.440-451.

[14] Maoz T., Barak A. and Amar L., Combining Virtual Machine Migration with Process Migration for HPC on Multi-Clusters and Grids, IEEE Cluster, Tsukuba, 2008.

[15] A Ganguly, A Agrawal, PO Boykin, and RJ Figueiredo, "WOW: Self-organizing Wide Area Overlay Networks of Virtual Workstations," Journal of Grid Computing, vol. 5, Apr. 2007, pp. 151-172.